TP8 – Segmentation par champ de Markov

Rappels de cours

La segmentation d'une image en niveaux de gris $x = (x_s)$ peut être effectuée par *classification supervisée*. On choisit un nombre N de classes, supposées gaussiennes, et on suppose connues les moyennes $(\mu_i)_{i\in\{1,\dots,N\}}$ et les variances $(\sigma_i^2)_{i\in\{1,\dots,N\}}$ des différentes classes. Le résultat est la configuration $\widehat{k} = (\widehat{k_s})$ qui maximise la probabilité a posteriori de la configuration $k = (k_s)$, sachant x. Or, d'après le théorème de Bayes :

$$p(K = k|X = x) = \frac{p(X = x|K = k)\, p(K = k)}{p(X = x)} \;\propto\; p(X = x|K = k)\, p(K = k) \tag{1}$$

L'hypothèse d'indépendance des données permet d'écrire la vraisemblance sous la forme d'un produit :

$$p(X = x|K = k) = \prod_{s\in\mathcal{S}} p(X_s = x_s|K_s = k_s) = \prod_{s\in\mathcal{S}} \frac{1}{\sqrt{2\pi}\,\sigma_{k_s}} \exp\left\{-\frac{(x_s - \mu_{k_s})^2}{2\,\sigma_{k_s}^2}\right\} \tag{2}$$

Quant à la probabilité a priori de la configuration k, elle est donnée par le *modèle de Potts* :

$$p(K = k) \;\propto\; \exp\left\{-\beta \sum_{\{s,t\}\in\mathcal{C}_2} [1 - \delta(k_s, k_t)]\right\} \tag{3}$$

où $\mathcal{C}_2$ désigne l'ensemble des cliques de cardinal 2, c'est-à-dire l'ensemble des paires $\{s, t\}$ de pixels voisins (avec le système de voisinage des « 8 plus proches voisins »). On déduit des équations (1), (2) et (3) :

$$p(K = k|X = x) \;\propto\; \exp\{-U(k)\} = \exp\left\{-\sum_{s\in\mathcal{S}}\left[\ln\sigma_{k_s} + \frac{(x_s - \mu_{k_s})^2}{2\,\sigma_{k_s}^2}\right] - \beta \sum_{\{s,t\}\in\mathcal{C}_2} [1 - \delta(k_s, k_t)]\right\} \tag{4}$$

Trouver le maximum de $p(K = k|X = x)$ équivaut donc à chercher le minimum de l'énergie $U(k)$. Pour ce faire, il ne suffit pas d'optimiser l'énergie localement, en chaque pixel $s \in \mathcal{S}$, ce qui s'écrirait :

$$\widehat{k_s} = \underset{k_s\in\{1,\dots,N\}}{\arg\min} \left\{\ln\sigma_{k_s} + \frac{(x_s - \mu_{k_s})^2}{2\,\sigma_{k_s}^2} + \beta \sum_{t\in\mathcal{V}(s)} [1 - \delta(k_s, k_t)]\right\} \tag{5}$$

On doit chercher le minimum *global* de $U(k)$. Or, il est impensable de tester les $N^{\mathrm{card}(\mathcal{S})}$ configurations possibles. On peut en revanche utiliser le *recuit simulé*. Cette *méta-heuristique* fait décroître un paramètre T, appelé *température*, en le multipliant par $\alpha < 1$ à chaque itération. L'algorithme complet s'écrit donc :

1. **Initialisations :** $T \leftarrow T_0$; $K \leftarrow$ Valeurs obtenues par maximisation de la vraisemblance.
2. **Parcours de tous les pixels s de l'image, visitée ligne par ligne et colonne par colonne :**
   - Tirer une nouvelle réalisation Nouv de K_s dans $\{1, \dots, N\}\backslash\{k_s\}$ et comparer les énergies locales :

$$\begin{cases} U_{\mathrm{Cour}} = \ln\sigma_{k_s} + \dfrac{(x_s - \mu_{k_s})^2}{2\,\sigma_{k_s}^2} + \beta \displaystyle\sum_{t\in\mathcal{V}(s)} [1 - \delta(k_s, k_t)] \\ U_{\mathrm{Nouv}} = \ln\sigma_{\mathrm{Nouv}} + \dfrac{(x_s - \mu_{\mathrm{Nouv}})^2}{2\,\sigma_{\mathrm{Nouv}}^2} + \beta \displaystyle\sum_{t\in\mathcal{V}(s)} [1 - \delta(\mathrm{Nouv}, k_t)] \end{cases} \tag{6}$$

   - Si $U_{\mathrm{Nouv}} < U_{\mathrm{Cour}}$, alors $K_s \leftarrow$ Nouv. Sinon, la nouvelle réalisation Nouv peut quand même être acceptée, avec une probabilité $\exp\{-(U_{\mathrm{Nouv}} - U_{\mathrm{Cour}})/T\}$ qui décroît avec la température T. Une particularité du recuit simulé est donc de ne pas systématiquement éliminer les changements de configuration qui font croître l'énergie.
3. **Mises à jour :** $T \leftarrow \alpha\, T$, puis retour en 2, tant que le nombre maximal $q_{\max}$ d'itérations n'est pas atteint.

Exercice 1 : segmentation d'une image en classes prédéfinies

Écrivez les fonctions `attache_donnees`, `regularisation` (modèle de Potts 8-connexe) et `recuit_simule`, de telle sorte que le script `exercice_1` segmente l'image `image.bmp` selon la méthode décrite ci-dessus :

- La fonction `attache_donnees` doit retourner une matrice tridimensionnelle contenant, pour chaque pixel, la valeur de l'attache aux données relativement à chacune des N classes. Attention : dans l'expression (5), σ_{k_s} désigne un écart-type et non pas une variance !
- La fonction `regularisation` doit retourner le terme de régularisation locale de l'expression (5). L'expression $[1 - \delta(k_s, k_t)]$ s'écrit très simplement en Matlab à l'aide de l'opération `~=` (« différent de »).
- Enfin, dans la fonction `recuit_simule`, le tirage aléatoire d'un nombre réel selon une loi uniforme dans l'intervalle $[0, 1]$ s'écrit tout simplement `rand`.

Dans cet exercice, les paramètres des classes de pixels (moyennes et écarts-types) sont prédéfinis. En revanche, les paramètres T_0, $q_{\max}$, α et β doivent être ajustés de façon à maximiser le pourcentage de bonnes classifications.

Exercice 2 : segmentation par classification supervisée

Faites une copie du script `exercice_1`, de nom `exercice_2`, que vous modifierez de manière à apprendre les paramètres des classes sur des échantillons sélectionnés par l'utilisateur (classification *supervisée*) : la fonction `estimation` permet de sélectionner des échantillons rectangulaires caractéristiques des différentes classes, en cliquant sur deux points, considérés comme les sommets opposés d'un rectangle.

Le calcul du pourcentage de pixels correctement classés s'appuie sur une carte de bonnes classifications contenue dans le fichier `classification_OK.mat`. Or, ce calcul n'est valide que si les classes sont triées par ordre croissant de la moyenne. Modifiez le script `exercice_2` de manière à rendre correct le calcul de ce pourcentage, en permutant les moyennes, les variances et les colonnes de la matrice `couleurs_classes`.

Les résultats doivent être comparables à ceux de l'exercice 1, lorsque $N = 6$ et que les échantillons des différentes classes sont correctement sélectionnés. Observez ce qui se passe dans les cas suivants : lorsque le nombre N de classes est différent de 6 ; lorsque les échantillons sont mal sélectionnés ; lorsque $T_0 = 0$, ce qui élimine tous les changements de configuration qui font croître l'énergie.

Exercice 3 : utilisation de la couleur

Il serait dommage de ne pas utiliser l'information de couleur, qui constitue un indice de première importance pour segmenter une image. Écrivez la fonction `attache_donnees_RVB`, appelée par le script `exercice_3`, dont deux des paramètres d'entrée permettent de tenir compte de la couleur : la matrice `moyennes`, de taille $3 \times N$, contient une couleur moyenne par colonne ; la matrice tridimensionnelle `variances_covariances`, de taille $3 \times 3 \times N$, est telle que `variances_covariances(:,:,c)` contient la matrice de variance-covariance de la classe `c`. Il est rappelé que, si $x \in \mathbb{R}^d$, la loi normale en dimension d s'écrit :

$$p(x) = \frac{1}{(2\pi)^{d/2}\sqrt{\det \Sigma}} \exp\left\{-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right\} \tag{7}$$

où $\mu \in \mathbb{R}^d$ et $\Sigma \in \mathbb{R}^{d\times d}$ désignent, respectivement, la moyenne et la matrice de variance/covariance.

Vous disposez maintenant d'une méthode de segmentation d'une image RVB quelconque, mais n'oubliez pas que cette méthode est *supervisée* : vous devez fixer par avance le nombre N de classes et sélectionner un échantillon par classe. Ce travail nécessite généralement l'intervention d'un « expert ». Vous pouvez vous prêter à ce jeu sur l'image `guadeloupe.jpg`, ou sur toute autre image de votre choix.

Exercice 4 : détection du bâti dans une image SPOT (facultatif)

Lancez le script `exercice_4`, qui affiche une petite partie d'une image SPOT et calcule, en chaque pixel, un « coefficient de texture » (notion intuitive dont la définition précise est relativement complexe). En vous inspirant des exercices précédents, modifiez ce script de manière à détecter les zones bâties, qui se caractérisent manifestement par un coefficient de texture plus élevé que dans les zones agricoles.